Acceleration of HE-Transformer with bit reduced SEAL and HEXL

Xinyi LI[†], Masaki NISHI[†], Teppei SHISHIDO[†], and Keiji KIMURA[†]

[†] Department of Computer Science and Engineering, Waseda University
E-mail: †xylee@toki.waseda.ac.jp, †masaki@kasahara.cs.waseda.ac.jp,
†teppei14860623@kasahara.cs.waseda.ac.jp, †keiji@waseda.jp

Abstract While cloud applications are covering people's privacy data, their related risks, such as theft risk, leak risk, and other serious risks, are becoming critical issues. Homomorphic encryption (HE) is a promising approach to overcome them since it enables calculation of encrypted data without decryption. Deep learning (DL) inference on cloud is an interesting application of HE. However, the computational cost of HE is too expensive, resulting in its limited usage in the real world. In this paper, we propose an acceleration technique of DL on HE, and implement it in HE-Transformer, which is a framework for DL on HE. Our approach is reducing the bit-width of internal data structure of HE libraries, SEAL and HEXL, from 64-bit to 32-bit. As a result, we can obtain up to 3.59× speedup compared to the original version, while still keeping data privacy.

Key words Homomorphic Encryption, Deep Learning, SIMD

1. Introduction

Variety of digital services is becoming important parts of our society. Particularly, Deep Learning (DL) based services have became inevitable since they can naturally support our daily lives considering various information surrounding us. They now also handle our confidential information, such as health information, financial information, and so on.

While the importance of DL services increase, the computing power required for training also increases as the size of datasets and neural networks become larger and the demand for prediction accuracy rises. Thus, it pushes most companies and organizations to train neural network models on cloud computing centers. Also, services based on DL inference utilizing the trained model tend to be deployed on the cloud. This situation causes the exposure of service users' privacy information to security risks, such as data steal and tampering, as well as the exposure of confidential companies' trained models.

Homomorphic encryption (HE) is an encryption method that enables encrypted data to perform mathematical operations [1]. Employing HE for DL on the cloud is a promising approach that can protect the users' privacy information and companies' confidential trained model. nGraph-HE (HE-Transformer), a backend of nGraph from Intel, supports the application of homomorphic encrypted data to deep learning, i.e., using encrypted data to do evaluations on neural network models [2]. Specifically, HE-Transformer supports CKKS encryption scheme, which enables calculations on real and complex numbers on HE [3]. However, such computation on ciphertext requires computing power and time that are often far more than those of plaintext. Even if it can be used for deep learning, its cost seems too expensive at present.

To get better performance on HE-Transformer, we have updated it from the original Ver.3.4.5 to a newer version that integrates HEXL [4]. Moreover, considering the still huge computing power requirement, we here apply our optimized new SEAL which provides a 32-bit version for the CKKS encryption scheme [5]. It enables speedup the computation of encrypted data while guaranteeing confidentiality. At the same time, the integration of HEXL makes it possible to utilize AVX512 [6], which further accelerates the computation. Reduced bit versions of SEAL and HEXL also make it possible to load more data on the AVX512 architecture at one time. As the result of updating to newer version of SEAL, the calculation is accelerated at most 1.98× on the evaluations based on MNIST dataset and ImageNet dataset. Then, applying bit-reduced SEAL and HEXL can obtain up to 2.89× of speedup. That is in total, the integration of bit-reduced SEAL has made HE-Transformer achieve 3.59× acceleration.

Other parts of this paper include: Section 2. descirbes the background of each technique. Section 3. summarizes related works. Section 4. describes details of the works. Section 5. shows the evaluation and the results. Finally Section 6. concludes our work.

2. Backgrounds

2.1 Homomorphic Encryption (HE)

Among the large number of encryption algorithms, homomorphic encryption (HE) can meet the need for encrypted data processing while maintaining confidentiality. It was co-proposed by Ronald L. Rivest and others in 1970s [1]. Differing from usual encryption algorithms, additions and multiplications can be processed on the ciphertext by HE as follows:

$$Dec(Enc(m_1) + Enc(m_2)) = Dec(Enc(m_1 + m_2)) \quad (1)$$

$$Dec(Enc(m_1) \times Enc(m_2)) = Dec(Enc(m_1 \times m_2)) \quad (2)$$

The idea of HE is based on Learning with Errors (LWE) from Lattice-based Cryptography [7]. Given $a_1, a_2, a_3, ..., a_n \in \mathbb{R}^m$ as a set of base vectors, and $A \in \mathbb{R}^{m\times n}$.

$$\mathcal{L}(A) = \mathcal{L}(a_1, a_2, a_3, ..., a_n) = \{Ax | x \in \mathbb{Z}^n\} \quad (3)$$

$\mathcal{L}(A)$ is made of discrete points. Then given equation:

$$\hat{b} = A \cdot x + e \quad (4)$$

Where error e is randomly chosen in specified range. When A and $\hat{b}$ are given, LWE aims to find out x that makes the above equation hold. The problem of LWE is NP-hard and it is difficult to distinguish vector $\hat{b}$ from random vector set v due to the interference of e. Such features ensure the security of HE.

The process of encryption and decryption of HE can be described as below, where P is plaintext, $pk = (-A \cdot s + e, A)$ is the public key and $sk = s$ is the secret key.

$$\begin{aligned} Encrypt(P, pk) &= (P, 0) + (-A \cdot s + e, A) \\ &= (-A \cdot s + e + P, A) \\ &= (c_0, c_1) = c \end{aligned} \quad (5)$$

$$Decrypt(c, sk) = c_0 + c_1 \cdot s = P + e \quad (6)$$

The plaintext will be encrypted into ciphertext $c = (c_0, c_1)$ by the public key pk. After decryption, the ciphertext is decrypted into $P + e$, , where e is a manually introduced noise in pk. While ensuring security, the error e also becomes the source of inaccuracy.

2.2 Cheon-Kim-Kim-Song (CKKS) Scheme

CKKS is a homomorphic encryption scheme aiming to arithmetic of approximate numbers. It develops besed on Ring Learning with Errors (RLWE), which utilizes polynomial rings instead of matrices [8]. Rather than encrypting integers, CKKS aims to process real and complex numbers.

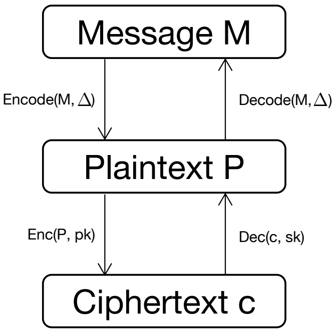

Figure 1 Overview of CKKS Scheme.

Figure 1 describes the CKKS scheme. It is similar to regular other RLWE based HE systems, such as BFV [9], except for the parameter of scale Δ. The message M, usually a single number or a vector of numbers, will be expanded by scale Δ at first, and then be encoded to a plaintext P in form of a ring polynomial $m(X) \in \mathbb{Z}[X]/\left(X^N + 1\right)$.

Similar to Section 2.1, ciphertext is in form of $c = (c_0, c_1)$, except that it is composed of two polynomials. And ciphertext will also be decrypted into $P+e$, where the introduced e is the reason why CKKS scheme is for arithmetic of approximate numbers. However, although the noise will affect the accuracy, this kind of effect is greatly diminished by scale Δ.

When two ciphertexts c and c' are added together:

$$\begin{aligned} c'' &= Add(c, c') = (c_0, c_1) + (c'_0, c'_1) \\ &= (c_0 + c'_0, c_1 + c'_1) \end{aligned} \quad (7)$$

The result of addition conforms to the form of the ciphertext and is capable of using `Decryption` function.

But when two ciphertexts are multiplied together, since ciphertext exists as form of polynomials, multiplication is much more complicated:

$$\begin{aligned} d = Multiply(c, c') &= (c_0, c_1) \times (c'_0, c'_1) \\ &= (c_0 \times c'_0, c_0 \times c'_1 + c_1 \times c'_0, c_1 \times c'_1) \\ &= (d_0, d_1, d_2) \end{aligned} \quad (8)$$

The result of multiplication d is composed of 3 polynomials, which is obvious that it can not be decrypted by `Decryption` function. Thus the new ciphertext needs `Relinearization`, by which the new ciphertext is relinearized down to normal form:

$$d' = Relin(d, Relin\_key) = (d'_0, d'_1) \quad (9)$$

Besides, `Rescaling` is used for scaling down the size of multiplied ciphertext. As the paper described before, message is encoded and encrypted with scale Δ. Without `Rescaling`, the scale of ciphertext will increase exponentially after multiplication. To realize such operation, CKKS scheme uses moduli $q = \Delta^L \cdot q_0$, where L indicates the number of capable multiplication and $\log_2(q_0) - \log_2(\Delta)$ indicates the bit of integer part.

Nevertheless, q is usually out of 64-bit word length that mainstream computers use, since multiple scales are accumulated. Chinese Remainder Theorem (CRT) is suitable for handling this situation by mapping a large number into a set of numbers *modulo* co-prime numbers.

Until now, CKKS scheme is used in large variety of HE libraries. For example, HElib supports BGV and CKKS cryptographic schemes [10], and SEAL from Microsoft supports BFV and CKKS cryptographic schemes [4].

2. 3 SEAL and HEXL

SEAL is an open-source HE library provided by Microsoft, which supports BFV and CKKS encryption schemes and also supports arithmetical operations on encrypted data [4]. HEXL is also an open-source library developed by Intel. The main purpose of it is accelerating frequently used calculations in HE [6]. As it develops so far, starting from Ver.3.6.3, SEAL has integrated HEXL Ver.1.1.0. Since then, SEAL can directly utilize HEXL without additional installation.

In our previous work, we have extended SEAL and HEXL so that they can process their calculation on 32bit internal data structure, instead of the original 64bit one [11]. The motivation of this work came from the well known insight that practical deep learning applications do not often 64bit-length for input data and parameters such as weights to train neural networks, as well as to achieve the required prediction accuracy [12]. By reducing the bit width of internal data structure, more SIMD width can be kept, resulting in higher performance.

2. 3. 1 Bit Reduced (32bit) SEAL

To achieve the bit reduced SEAL, the coefficient type of the polynomial expressing the ciphertext needs to be changed from 64bit to 32bit. This target can be attained by changing the `coeff_modulus` parameter in SEAL. Besides, the functions such as encoding, decoding, encryption, decryption, and ciphertext itself computations of SEAL need to be modified to use 32bit data structure. These functions not only change the type of their parameters and local variables but also the whole process of computation will be based on 32bit.

To further accelerate the ciphertext computation, in addition to the bit reduced SEAL, the generation of Galois key is also parallelized. Galois key is used for rotating the ciphertext on the CKKS encryption scheme, where the rotation is necessary for ciphertext multiplication. We have parallelized the loops for Galois key generation hierarchically by OpenMP and AVX512; outer loops are parallelized by OpenMP and inner additions and multiplications are vectorized by AVX512.

2. 3. 2 Bit Reduced (32bit) HEXL

HEXL is an open-source acceleration library from Intel. It enables acceleration of polynomial computation by leveraging the new instruction set AVX512 and applying it to Number-Theoretic Transform (NTT) [6]. This is feasible for SEAL, where both BFV and CKKS are encrypted based on polynomials.

In the current version of SEAL, HEXL has been already integrated. With the implementation of the bit reduced SEAL, HEXL needs to be changed accordingly.

For the development of the bit reduced HEXL, since is is built based on the AVX512 instruction set, these AVX512 instructions in it are also modified to use 32bit data structure. For example, in the function that performs multiplication to obtain the low bit, instead of using `_mm512_mullo_epi64`, we use the `_mm512_mullo_epi32` instruction [11]. Considering the overall feasibility, there are still plenty of modifications besides changes to the instructions.

3. Related Works

Much of the known work also deals with how to apply HE for deep learning [5], [13]〜[15], but they mainly focus on algorithm-level optimization or acceleration. This paper concentrates on data structures. And additionally, our work can be applied to the papers mentioned before, leading to better optimization.

4. Adapting Bit Reduced SEAL and HEXL to HE-Transformer

HE-Transformer aims to accelerate processing of HE enabled neural networks. It works as a backend of nGraph, which is a DL compiler. As shown in Figure 2, it takes a graph intermediate representation (IR) from nGraph, and utilizes SEAL to process add, multiply, convolution, and other DL related operations on HE cryptsystems, such as CKKS and BFV, as well as employing HE related optimizations. This section describes how we adapted the bit reduced SEAL and HEXL to HE-transformer.

4. 1 Modification for Updated SEAL

As of today, the latest update of HE-Transformer is dated 2020. The version of SEAL used by it was 3.4.5. Compared to the existing version 3.6.6, where two major version updates were made, there is a considerable gap between two versions, both in terms of performance and code. In particular, the original version did not introduce HEXL and could not take advantage of the AVX512 instruction set. Therefore, we first adapted SEAL 3.6.6 to HE-Transformer instead of the originally used 3.4.5.

First, we investigated the differences of class hierarchy

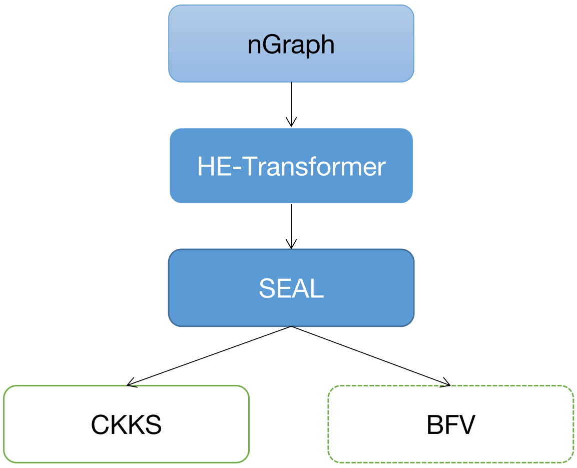

Figure 2　Overview of HE-Transformer. This paper mainly focuses on CKKS sheme.

between two versions of SEAL, then we modified HE-Transformer to adapt to newer class hierarchy. For example, SEAL 3.4.5 had the class `SmallModulus`, and, by 3.6.6, it has been merged into `Modulus`, which causes problems with incorrect calls in HE-Transformer.

Additionally, several functions are changed to newer ones. A function of key-generation is such an example. In SEAL 3.4.5, the public key was generated by the function `KeyGenerator::public_key()`, while `KeyGenerator::create_public_key()` has the same function in the version 3.6.6.

Meanwhile, the treatment of `SEALContext`, which contains encryption parameters, has also been changed. Unlike the old version where a generated `SEALContext` is passed in form of a pointer, the new version generates a `SEALContext` and passes it as an instance directly, resulting in the modification of all its related code.

Moreover, the name of scheme type is also changed from `CKKS` to `ckks` using lower letters.

Besides, for the thought of the integrity, modifying codes needs to be as much as carefully considered.

4. 2　Modification for Bit Reduced SEAL

As stated in Section 2. 3, the bit reduced SEAL and HEXL introduce 32bit data structure in all aspects. Accordingly, HE-Transformer has to be modified to use 32bit data structure for the relevant parts as well. For example, all parameters with the original data type `uint_64` is changed to `uint_32`, and the `double` type is changed to `float`, and so on. Simultaneously, the potential data overflow should be considered due to the change of data types.

In addition, the functions involved must be changed according to the 32-bit SEAL.

For instance, `multiply_uint64_hw64` is changed to `multiply_uint32_hw32`, which has been improved in the bit reduced SEAL, and `barrett_reduce_128` is changed to `barrett_reduce_64`.

Table 1　Machine Specification

CPU	Intel Xeon W-2145
Number of Cores	8
Base Frequency	3.7GHz
Support AVX512	Yes
L1D Cache	32KiB/core
L2 Cache	1MiB/core
L3 Cache	11MiB
Main Memory	94GiB
OS	Ubuntu 20.04

5.　Evaluation

This section shows the performance evaluation of the accelerated HE-Transformer based on the premise of the modifications described in Section 4.

5. 1　Evaluation Environment

We conducted the evaluation on an Intel Xeon W-2145 machine shown in Table 1. We evaluated CryptoNets with MNIST and MobileNet with ImageNet, which are also included in HE-Transformer.

5. 2　Parameter Setting

In order to ensure that the evaluation can be performed on different versions and that the evaluation results can be compared directly. In this paper, the parameters of the original version, the 64-bit version and the 32-bit version are all set uniformly. `coeff_modulus` and `scale` are both set based on 29 bits with security promised as well. In particular, the length of `coeff_modulus` decides the number of multiplication encrypted data can perform. `scale` is the parameter used to expand the input message, and it should be the power of 2.

5. 3　Evaluation Result

5. 3. 1　MNIST Evaluation

CryptoNets is a widely used framework in the field of HE [13]. It supports converting trained neural network model into networks that is available for HE data. We here utilize Cryptonets and do evaluations based on Modified National Institute of Standards and Technology (MNIST) dataset [16].

The configuration in Listing 1 is used for this evaluation. `"scheme_name":"HE_SEAL"` means we use `SEAL` backend here. `poly_modulus_degree` affects the maximum number of value HE-Transformer can encrypt at one time. It should be noted that `complex_packing` decides whether packing every two real numbers into one complex number at the encoding. Here we set `batch_size` to 8192.

Figure 3 depicts the result of evaluation on MNIST dataset, where Complex Packing means the result after acti-

Listing 1 Parameters for MNIST Evaluation

```
{
   "scheme_name": "HE_SEAL",
   "poly_modulus_degree": 16384,
   "security_level": 128,
   "coeff_modulus":
       [29,29,29,29,29,29,29,29,29,29],
   "complex_packing": true,
   "scale": 536870912
}
```

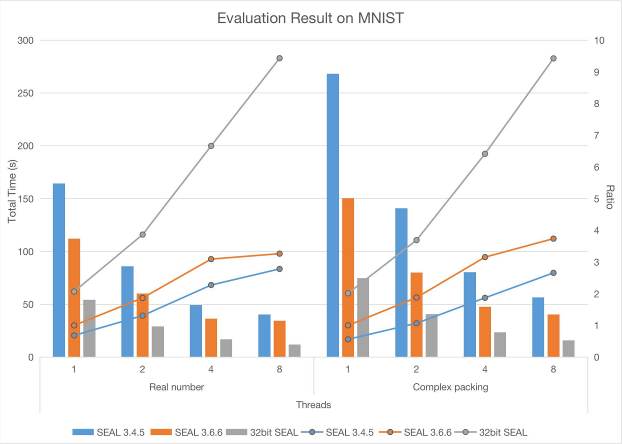

Figure 3 Evaluation Result on MNIST

Table 2 Speedup After Integrating 32bit SEAL (MNIST) for Each Number of Threads

	Threads							
	Real number				Complex packing			
	1	2	4	8	1	2	4	8
vs. 3.6.6	2.07	2.07	2.16	2.89	2.01	1.96	2.03	2.52
vs. 3.4.5	3.03	2.96	2.93	3.39	3.59	3.46	3.43	3.55

vating `complex_packing`. The entire test is conducted in three rounds, and then we take the average of the three rounds as our final test result. The result shows the performance improvement of the bit reduced 32bit SEAL comparing with the original 3.4.5. Take the performance of Ver.3.6.6 in the case of single-thread as our benchmark, the 32bit HE-Transformer attains 2.01×–9.43× speedup with the increasing number of threads. As shown on Table 2, at comparing the 32bit version with original one in each number of thread, we achieve at most 3.59× acceleration. Then compared to SEAL 3.6.6 version, the 32bit version still achieves at most 2.89× acceleration. Moreover, the evaluation can still achieve 98.60% classification accuracy, only about 0.5% lower than non-encrypted cases. `complex_packing` indeed decreases the speed, whereas it will double the capacity of input. Therefore in terms of the processing speed of a single image, the result is well accepted.

Listing 2 Parameters for ImageNet Evaluation

```
{
   "scheme_name": "HE_SEAL",
   "poly_modulus_degree": 4096,
   "security_level": 0,
   "coeff_modulus": [29,29,29,29,29],
   "complex_packing": true,
   "scale": 536870912
}
```

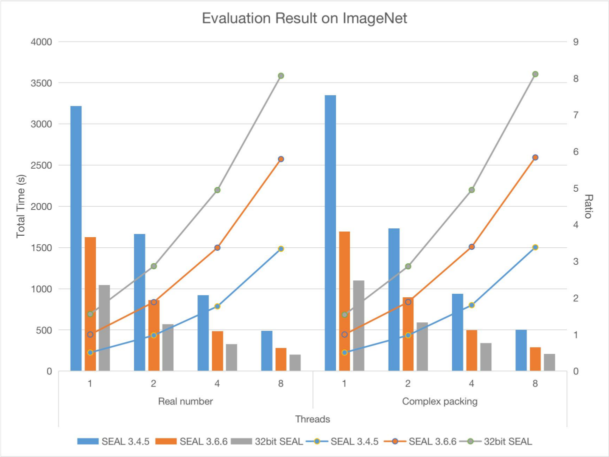

Figure 4 Evaluation Result on ImageNet

Table 3 Speedup After Integrating 32bit SEAL (ImageNet) for Each Number of Threads

	Threads							
	Real number				Complex packing			
	1	2	4	8	1	2	4	8
vs. 3.6.6	1.56	1.52	1.47	1.39	1.54	1.52	1.46	1.39
vs. 3.4.5	3.08	2.93	2.80	2.42	3.04	2.92	2.75	2.40

5.4 ImageNet Evaluation

Here we test HE-Transformer by using MobileNet to make inference, with dataset from ImageNet [17]. Encryption parameters are set as Listing 2.

Due to the larger size of the neural network, increased scale of dataset and limitation of memory, this paper chooses relatively small encryption parameters here, where the security level is set to 0, only for comparison of performance. We set `batch_size` to 512 for this evaluation.

Figure 4 depicts the result of the evaluation on it. Similar to the evaluation on MNIST dataset, we conduct three rounds of tests and choose the average result of them. As the Table 3 reveals, after integrating the 32 bit SEAL, the runtime is accelerated to 3.08× compared to original HE-Transformer, while still maintain the accuracy at 67.58%, which is similar to evaluation of plaintext. The figure shows that, using single-threaded Ver.3.6.6 as baseline, we have achieved 1.54×–8.11× of acceleration along with increasing the number of threads. Table 3 also shows, at comparing the 32bit version with original one in each number of thread, we achieve at most 3.08× acceleration. Even if compared to Ver.3.6.6 in different number of thread, the 32bit version can achieves up to 1.56× acceleration.

6. Conclusion

In this paper, we realized acceleration of HE-Transformer for DL processing on HE. Modifications were implemented in two steps, including the updating from Ver.3.4.5 to Ver.3.6.6 and integrating the bit reduced SEAL.

As the result of acceleration, in case of single thread, we achieve 3.59× and 3.08× speedup on MNIST and ImageNet datasets with different neural networks with comparable accuracy with the calculation on plaintext .

Reference

[1] R.L. Rivest and M.L. Dertouzos, "On data banks and privacy homomorphisms," 1978.

[2] F. Boemer, A. Costache, R. Cammarota, and C. Wierzynski, "ngraph-he2: A high-throughput framework for neural network inference on encrypted data," 2019. https://arxiv.org/abs/1908.04172

[3] F. Boemer, Y. Lao, R. Cammarota, and C. Wierzynski, "ngraph-he: A graph compiler for deep learning on homomorphically encrypted data," 2018. https://arxiv.org/abs/1810.10121

[4] "Microsoft SEAL (release 3.6)," `https://github.com/Microsoft/SEAL`, Nov. 2020. Microsoft Research, Redmond, WA.

[5] L.T. Phong, Y. Aono, T. Hayashi, L. Wang, and S. Moriai, "Privacy-preserving deep learning via additively homomorphic encryption," IEEE Transactions on Information Forensics and Security, vol.13, no.5, pp.1333–1345, 2018.

[6] F. Boemer, S. Kim, G. Seifu, F.D.M. deSouza, and V. Gopal, "Intel hexl: Accelerating homomorphic encryption with intel avx512-ifma52," 2021. https://arxiv.org/abs/2103.16400

[7] D. Micciancio and O. Regev, "Lattice-based cryptography," pp.147–191, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. https://doi.org/10.1007/978-3-540-88702-7_5

[8] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," Cryptology ePrint Archive, Paper 2012/230, 2012. `https://eprint.iacr.org/2012/230`. https://eprint.iacr.org/2012/230

[9] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," Cryptology ePrint Archive, Paper 2012/144, 2012. `https://eprint.iacr.org/2012/144`. https://eprint.iacr.org/2012/144

[10] S. Halevi and V. Shoup, "Algorithms in helib," Cryptology ePrint Archive, Paper 2014/106, 2014. `https://eprint.iacr.org/2014/106`. https://eprint.iacr.org/2014/106

[11] S. Shishido, M. Nishi, X. Li, and K. Kimura, "Acceleration of homomorphic encryption library seal by reducing the number of arithmetic bits," Technical Report of IEICE (CPSY), vol.IEICE-121, no.425, pp.91–96, mar 2022.

[12] M. Courbariaux, Y. Bengio, and J.-P. David, "Training deep neural networks with low precision multiplications," 2014. https://arxiv.org/abs/1412.7024

[13] R. Gilad-Bachrach, N. Dowlin, K. Laine, K. Lauter, M. Naehrig, and J. Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," Proceedings of The 33rd International Conference on Machine Learning, eds. by M.F. Balcan and K.Q. Weinberger, vol.48, Proceedings of Machine Learning Research, PMLR, New York, New York, USA, 20–22 Jun 2016. https://proceedings.mlr.press/v48/gilad-bachrach16.html

[14] T. Ishiyama, T. Suzuki, and H. Yamana, "Highly accurate cnn inference using approximate activation functions over homomorphic encryption," 2020. https://arxiv.org/abs/2009.03727

[15] R. Dathathri, O. Saarikivi, H. Chen, K. Laine, K. Lauter, S. Maleki, M. Musuvathi, and T. Mytkowicz, "Chet: An optimizing compiler for fully-homomorphic neural-network inferencing," Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Association for Computing Machinery, New York, NY, USA, 2019. https://doi.org/10.1145/3314221.3314628

[16] L. Deng, "The mnist database of handwritten digit images for machine learning research," IEEE Signal Processing Magazine, vol.29, no.6, pp.141–142, 2012.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," 2009 IEEE conference on computer vision and pattern recognitionIeee 2009.